

# **Linear algebra algorithms for Betti numbers and torsions**

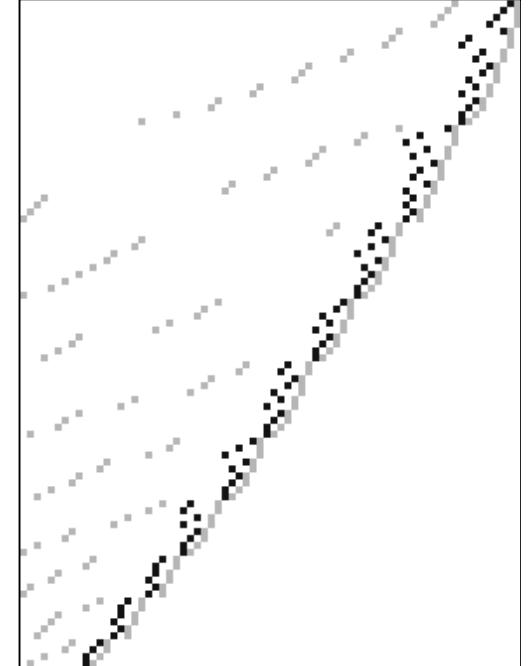
Jean-Guillaume Dumas

Université de Grenoble  
University College Dublin

## Computing Homology groups of simplicial complexes using linear algebra

Associate a matrix to each face (dim. N) and edge (dim. N-1):

$b_{ij} = \pm 1$  whenever edge j belongs to face i.



- $B_i = (b_{ij})$  = Boundary Matrix
- $i^{\text{th}}$  homology group  $H_i = \text{Ker } B_i / \text{Span } B_{i+1}$
- Smith diagonal form of  $B_i$ :  $S_i = \text{diag}(s_1, \dots, s_r, 0, \dots, 0)$   
[Smith]  $B \in \mathbb{Z}^{m \times n}$ , then  $\exists$  U and V unitary such that,  $B = U S V$
- Theorem : Let k be the first index such that  $s_k \neq 1$

$$H_i = \mathbb{Z}^{\text{rowdim}(B_i) - \text{rank}(B_i) - \text{rank}(B_{i+1})} \bigoplus \frac{\mathbb{Z}}{s_k \mathbb{Z}} \dots \bigoplus \frac{\mathbb{Z}}{s_r \mathbb{Z}}$$

# Algorithmic problems

- (1) ***Exact*** resolution : arbitrary size integers
  - ⇒ Work within several finite fields and extensions
  - ⇒ p-adic lifting, chinese remaindering
  - ⇒ Requires efficient finite field routines
  
- (2) ***Large Sparse*** linear systems
  - ⇒ Adapt sparse numerical methods
  - ⇒ Algorithms might have no numerical counterparts
    - ⇒ Rank computation, minimal polynomial, determinants ...
    - ⇒ Normal forms

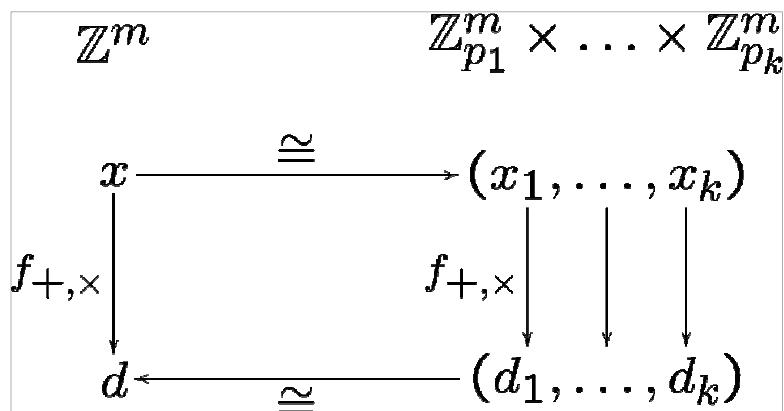
# Some issues

- Example = graph complex:  
**13** nodes complete graph matching complex
  - matrix **135 135 x 270 270**
  - Only 800 000 non-zero coefficients = 3 MBytes
- Over  $\mathbb{Z}$  we have a memory problem:
  - Elimination
    - Fill-in (up to  $n^2 = 36$  billions of non-zero coefficients)
    - Coefficient growth
      - naively exponentially up to  $2^n$  bits ...
      - still using modular methods and Hadamard bound up to  $n$  bits  
= 150000 GBytes)
  - ⇒ Rank: Compute modulo a prime not dividing the determinant
  - ⇒ Determinant, system solving: lifting/CRT
  - ...

# Integer coefficient growth

$A := [[2, 1, -3, 7, -4, 0, 9, 1, 6, -8, 6, -2, -8, -7, 4, -5, 5, 2, 4, -9, -1, -7, 2, 9, 2, 3, -1, 6, -8, 7, 3, -2, 0, -4, 8, 1, 2, 4, 6, 6, -4, -6, -3, -7, -8, 2, 4, -1, -2, -9, 8, 7, 6, -3, -9, 3, 4, 6, 3, 6, -6, -8, 3, -3, -1, -5, 6, 2, 2, -5, -5, -2, -9, -6, -1, -3, 5, 8, 8, -1, 2, 3, 8, -9, 3, -1, 8, -6, 7, -7, -8, 3, 7, 4, 3, 2, 2, -9, -2, -4]];$ , and  $b := [-2, -9, -3, 8, 0, -2, -6, 8, 5, -7]$ ;  
 $\rightarrow$  Solution of  $Ax=b$ :  
 $x = [-154378467, 166174189, 556790464, 72535956, 258019417, 499902615, -930948611, 1007780694, 356546217, -1001276698] / 55181746$

# Chinese remaindering, Hensel lifting

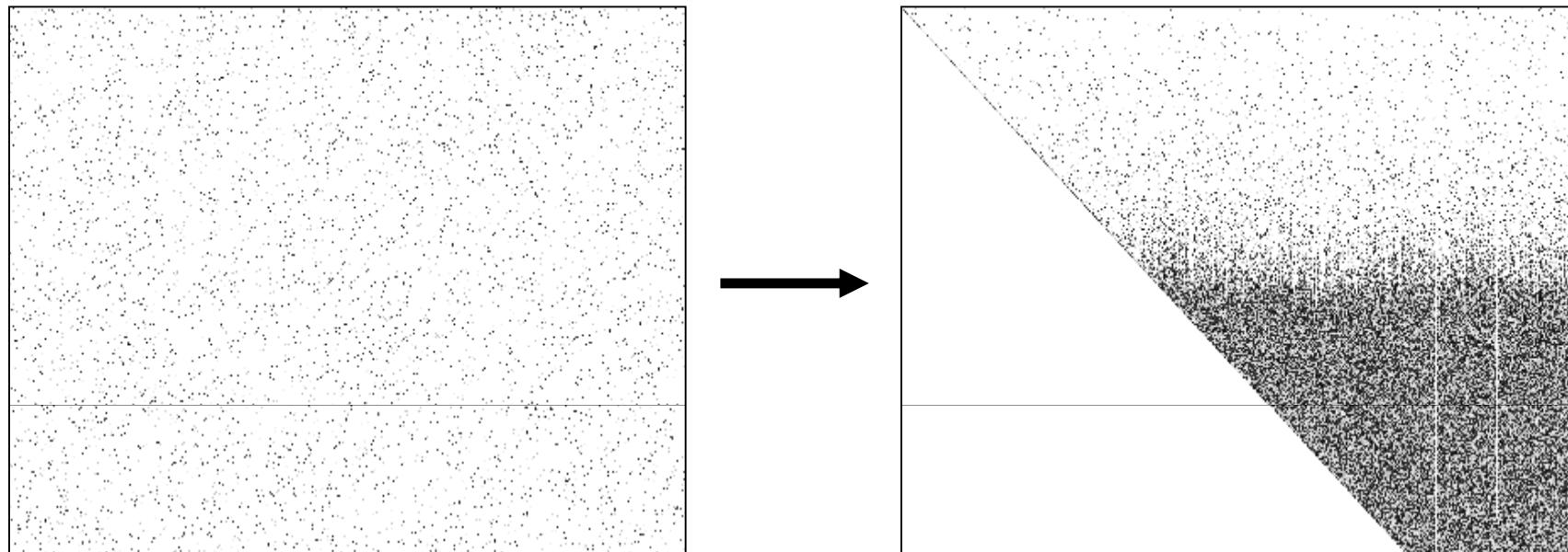


$O(Pb \times k + mk^2)$ , instead of  $O(Pb \times k^2)$

1. PLUQ(A) mod p  $n^w$
2. Iterate k times ( $k < n/2 \log(n)$ )
  - 2 triangular solve  $Ax_i \equiv b_i \pmod{p}$   $2n^2$
  - $b_{i+1} = (b_i - Ax_i)/p$  over  $\mathbb{Z}(n)$   $4n^2 \log(n)$
3.  $x = x_0 + p x_1 + p^2 x_2 + \dots + p^k x_k$   $k^2$
4. Rational reconstruction  $nk^2$

$O(n^w + n^3 \log^2(n))$ , instead of  $O(n^{w+1} \log(n))$

# Direct or iterative methods ?



Elimination fill-in: Fast at start, Fat at last

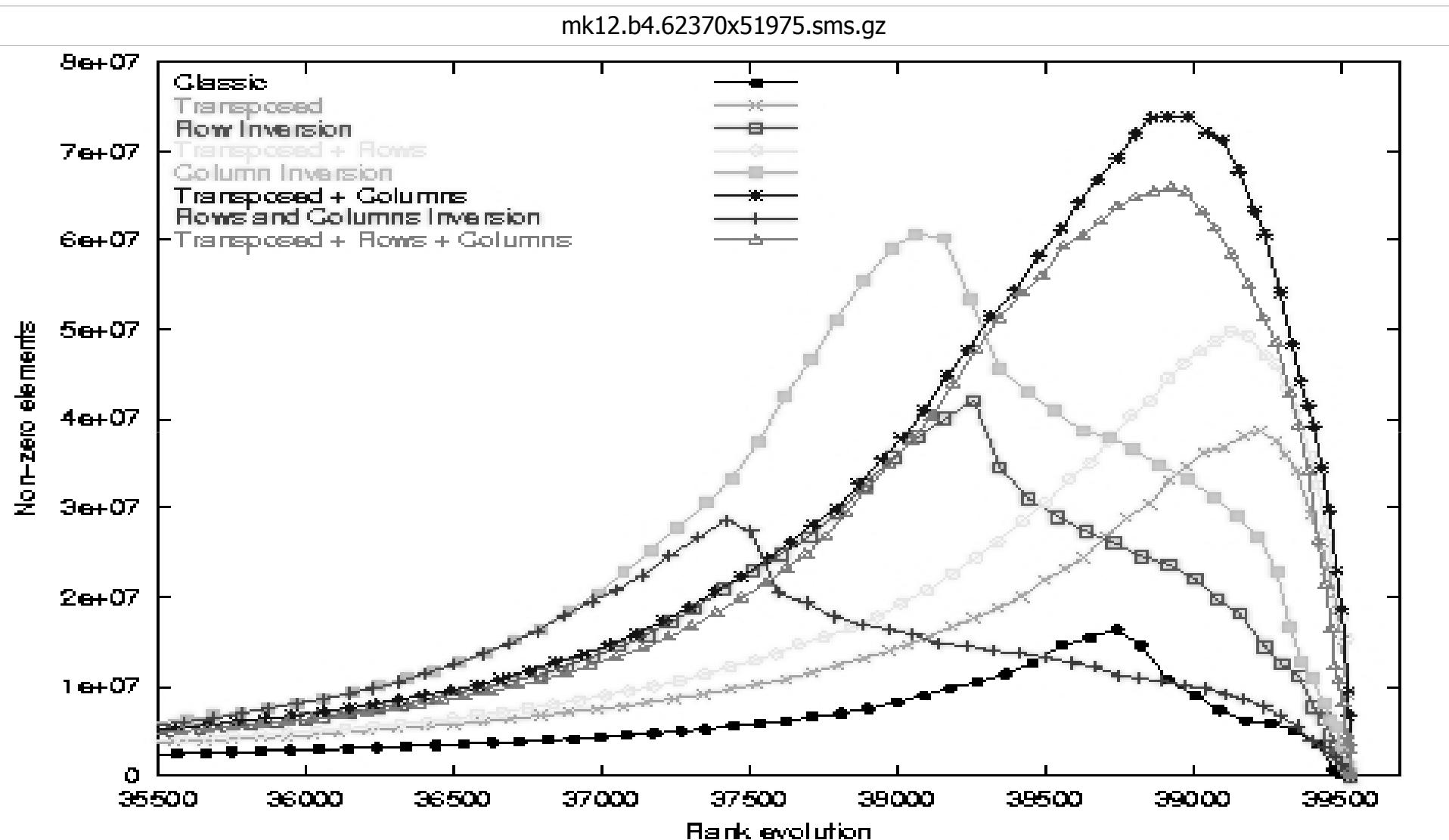


Blackboxes: Slowly but Surely

# **Contents**

1. Homology and linear algebra
2. Betti numbers
  - Rank via sparse elimination and reordering
  - Blackbox methods
  - Adaptive rank algorithm and domains of application
  - Parallel Wiedemann algorithm and scalability
3. Torsions
  - Sparse elimination with large precision integers and gcd's
  - Valence/Large invariant factor algorithm uses ranks
  - Adaptive Smith form

# Sparse elimination: fill-in problem



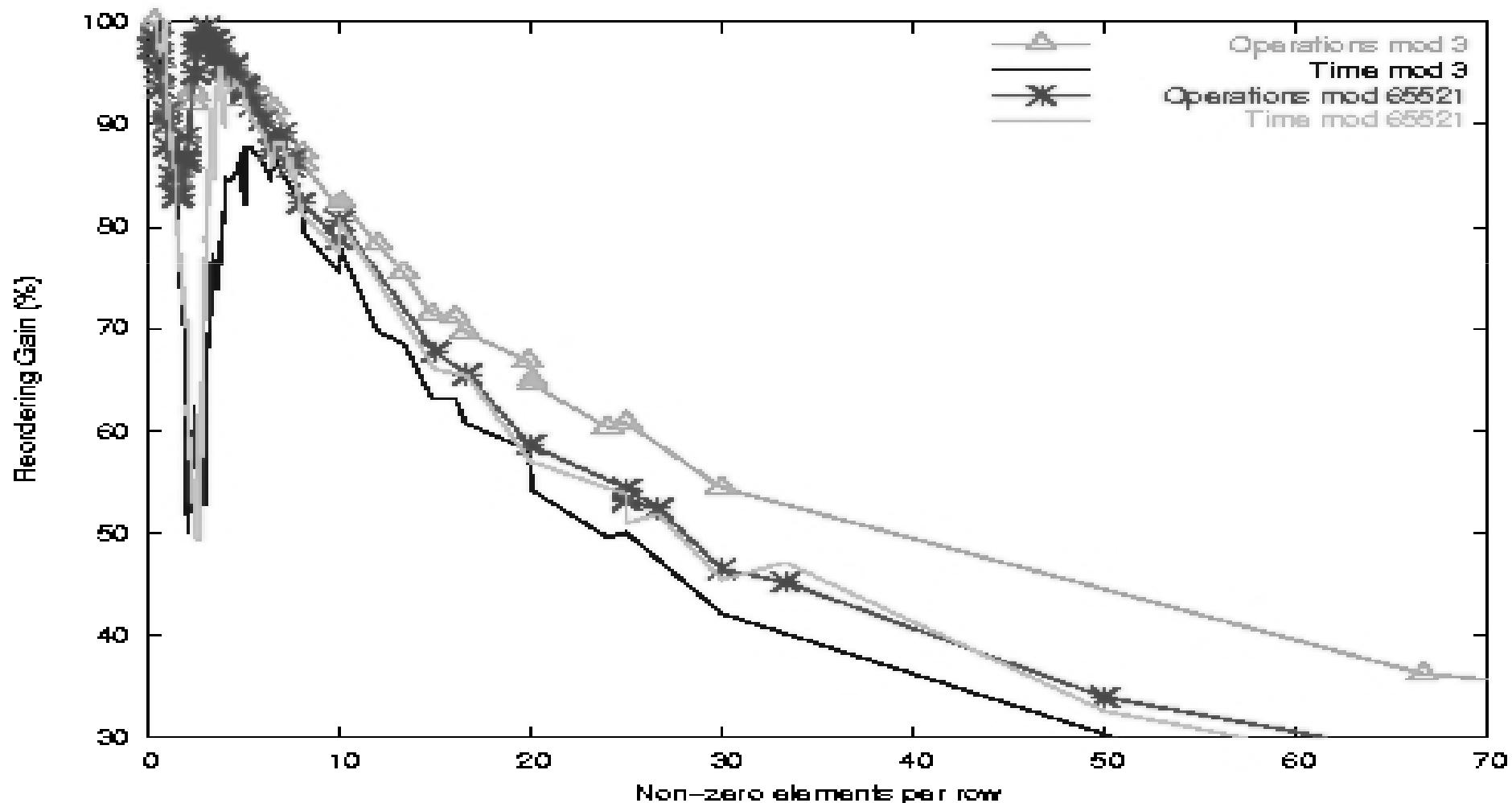
Heuristics: delay or reduce the dense phase to avoid memory thrashing:  
 $(62370-37000)*(51975-37000)*4*2 = 10 \text{ Gbytes}$

# Sparse Gaussian Elimination

- Major problem : fill-in
  - ⇒ memory thrashing
- Solutions : choose a pivot reducing fill-in  $\Leftrightarrow$  reorder
  - ⇒ minimal reordering : NP-complete ! *[Yannakakis 81]*
  - ⇒ reordering heuristics (from numerical methods):  
minimal degree ordering, nested dissections, *Markowitz'* cost function, ...
  - ⌚ Total pivot :  $O(n^2)$  overhead
- Computer algebra : structured gaussian elimination  
*[LaMacchia-Odlyzko 91]*
  - ⌚ Works on very over-determined systems
  - ⌚ Ends when too many rows are heavy

# Slightly intrusive heuristic [D.-Villard 2002]

- 1 Simplified LaMacchia-Odlyzko : singletons, pairs
  - 2 Simplified Markowitz : minimize  $r(i)$ , *then*  $c(j)$  in the chosen row
- ⇒ generic linear overhead + suited to small finite fields



# Experimental comparisons

Matrix	SuperLU	Markowitz	Light LO+M
cyclic8_m11	448.38	448.39	<b>257.33</b>
bibd_22_8	594.29	255.84	<b>100.24</b>
n4c6.b12	1312.27	640.39	<b>188.34</b>
Ch7-7.b5	MT	9204.48	<b>2179.62</b>
Ch7-8.b5	MT	24614.29	<b>5375.76</b>
TF13	3.54	6.95	<b>3.18</b>
TF14	<b>50.34</b>	83.98	50.58
TF15	776.68	2515.63	<b>734.39</b>
TF16	<b>15625.79</b>		18559.40

PIII, 1GHz, 1Gb

# **Contents**

1. Homology and linear algebra
2. Betti numbers
  - Rank via sparse elimination and reordering
  - Blackbox methods
  - Adaptive rank algorithm and domains of application
  - Parallel Wiedemann algorithm and scalability
3. Torsions
  - Sparse elimination with large precision integers and gcd's
  - Valence/Large invariant factor algorithm uses ranks
  - Adaptive Smith form

# Krylov Methods

Idea: only use the matrix-vector product

⇒ Matrix is never modified (Black box): constant memory usage

- Numerical Iterative methods
    - ⇒ fast convergence: a few iterations suffices
    - ⇒ several variants: Lanczos, conjugate gradient, Schmidt orthogonalization, ...
  - Exact computations:
    - ⌚ O( $n$ ) iterations
    - ⌚ Same number of iterations, but second-order complexity differs
- ⇒ Fastest variant ?
- |                     |           |                      |
|---------------------|-----------|----------------------|
| – System solving:   | Lanczos   | [Eberly-Kaltofen 97] |
| – Rank computation: | Wiedemann | [D.-Villard 02]      |

# Wiedemann overview

Minimal polynomial of  $A$ :  $\Pi(A)=0$ ,  $\text{degree}(\Pi)$  minimal  
 $\Rightarrow$  find the smallest relation between powers of  $A$ :  $\sum a_i A^i = 0$

- Scalar sequence by projection on a random vector:  
 $[ u^T u; u^T A u; \dots; u^T A^k u ] = [ s_0; s_1; \dots; s_k ]$
- Those scalars are linearly generated :  
 $b_0 s_0 + b_1 s_1 + \dots + b_k s_k = 0$
- Berlekamp/Massey computes  $b_i$  in a field via two growing polynomials  $\varphi$  and  $\psi$ , with high probability  $b_i = a_i$ :

- Repeat  $2d$  times
  - a) Discrepancy  $\delta_k = s_k + \varphi * S$   $k$  ops
  - b) Polynomial update  $\varphi = \varphi - \delta_k / \delta_{k+1}$   $X^e \psi \quad || \quad \psi = \varphi$   $k$  ops
  - c)  $v = A u$   $\Omega$  ops
  - d)  $s_{k+1} = u^T v$   $2n$  ops

Overall:  $2d\Omega + 2d(2d+n+m)$ , symmetric case:  $d\Omega + 4d(d+n)$

# Rank via minimal polynomial ?

- Idea: if  $A$  is diagonalizable =  $\text{diag}(\lambda_1, \dots, \lambda_r, 0, \dots, 0)$  then  
⇒ its characteristic polynomial is  $x^k \cdot \prod (x - \lambda_i)$   
⇒ rank is  $r=n-k$ .
- ☺ precondition [*Chen et al. 02*]:  
for random diagonal matrices  $D_1$  and  $D_2$ , with high probability,  
eigenvalues of  $D_1 B D_2 B^T D_1$  are distinct and its characteristic  
polynomial is  $x^k$ .(minimal polynomial).
- ☺ Monte-Carlo Algorithm :  
degree of minimal polynomial of  $(D_1 B D_2 B^T D_1)$   
is always a lower bound for the rank

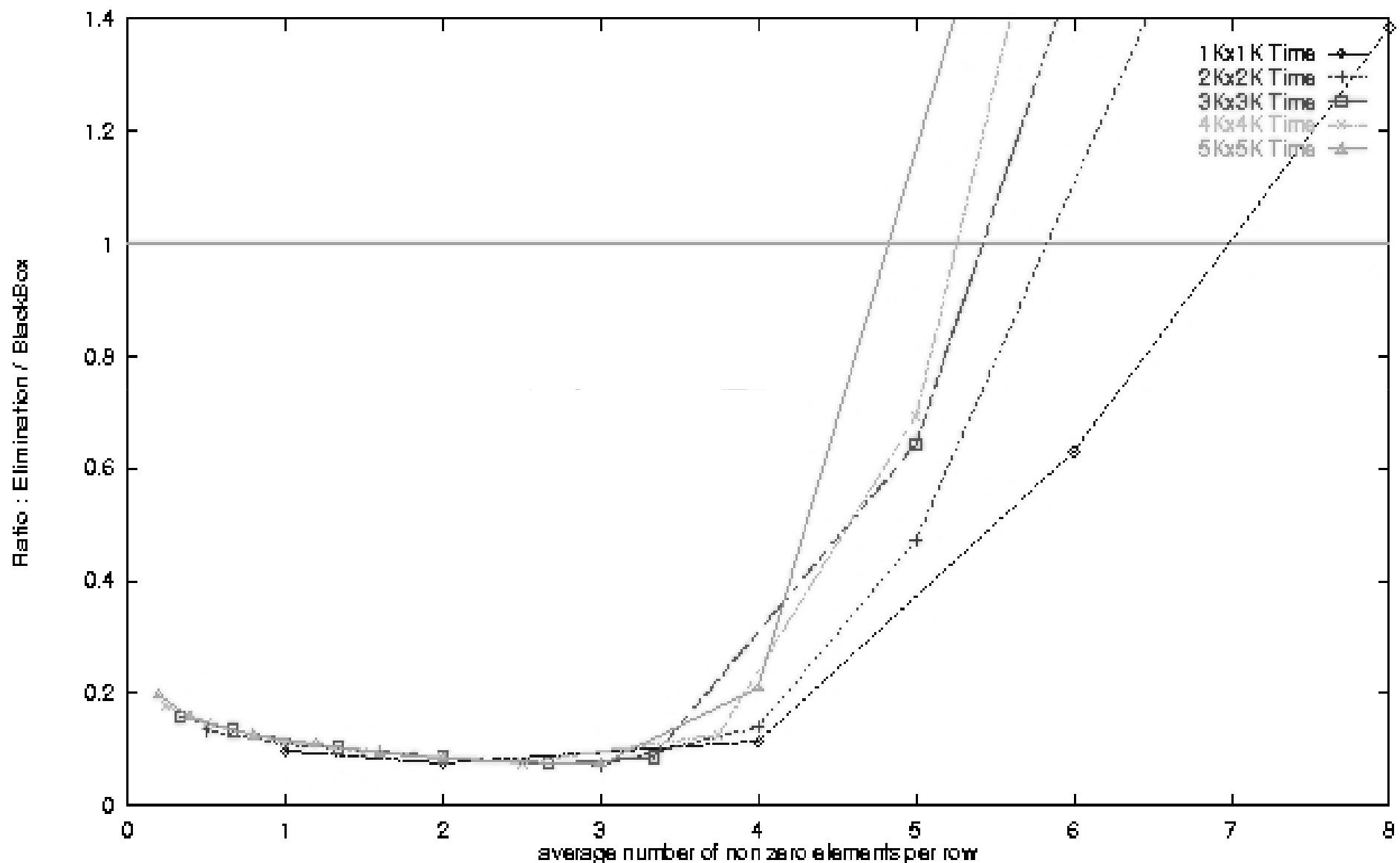
# Heuristics

- Minimal polynomial of  $(D_1 B D_2 B^T D_1)$ 
  - ☺ Works if field is sufficiently large (  $O(\log(n))$  ) [Eberly-Kaltofen'97]
    - 👉 First try over a word-size extension and perform fast checks
- Checks
  - [Saunders-Storjohann-Villard 04]
    - Monte-Carlo certifications of the rank
    - E.g.: trace is second coefficient of charpoly
  - Minpoly lives in the base field
  - lcm of minpoly with different vector projection should converge
  - ...
- Open problem: *fast* certification of the rank, a posteriori

# **Contents**

1. Homology and linear algebra
2. Betti numbers
  - Rank via sparse elimination and reordering
  - Blackbox methods
  - Adaptive rank algorithm and domains of application
  - Parallel Wiedemann algorithm and scalability
3. Torsions
  - Sparse elimination with large precision integers and gcd's
  - Valence/Large invariant factor algorithm uses ranks
  - Adaptive Smith form

# Ratio for Random matrices



# Homology Matrices

Matrix	<b>mk9 b3</b>	<b>ch7-7 b6</b>	<b>ch7-6 b4</b>	<b>ch7-7 b5</b>	<b>ch8-8 b5</b>	<b>n2c6 b6</b>	<b>n2c6 b7</b>	<b>n4c5 b6</b>	<b>n4c6 b12</b>
<b># rows</b>	<b>945</b>	<b>52920</b>	<b>15120</b>	<b>35280</b>	<b>564480</b>	<b>5715</b>	<b>3990</b>	<b>4735</b>	<b>25605</b>
<b>Non-zero</b>	<b>3780</b>	<b>35280</b>	<b>75600</b>	<b>211680</b>	<b>3386880</b>	<b>40005</b>	<b>31920</b>	<b>33145</b>	<b>1721226</b>
<b>Gauß</b>	<b>0.26</b>	<b>4.67</b>	<b>49.32</b>	<b>2179.62</b>	<b>Memory Thrashing</b>	<b>6.44</b>	<b>3.64</b>	<b>2.73</b>	<b>231.34</b>
<b>Wiedemann</b>	<b>2.11</b>	<b>119.53</b>	<b>416.97</b>	<b>4283.40</b>	<b>55 hours</b>	<b>72.96</b>	<b>57.10</b>	<b>51.75</b>	<b>4131.06</b>

Sequential ranks modulo 65521 on on a Sparc Ultra II 250 MHZ,  
 On Homology matrices [D, Heckenbach, Saunders, Welker 2003]

- mk  $i$ : complex from all the matchings with  $i$  vertices
- ch  $i-j$ : complex from the non taking rook positions on a  $i \times j$  chessboard
- n/c  $j$ : complex from the « not  $i$ -connected graph on  $j$  vertices »
- Constant number of non-zero elements per row : 3,4,5,6,7

# More Homology Matrices

matrix	M05-D4	M05-D5	M05-D6	M05-D7	M05-D8	M06-D5	M06-D6
size	210x2112	2112x7260	7260x11280	8160x11280	2240x8160	3024x49800	49800x294480
rank	210	1902	5358	5916	2239	3024	46776
Gauß	<10ms	0.15	2.25	2.17	0.18	0.51	369.03
Wiedemann	0.01	20.29	87.14	83.71	17.91	592.67	44724.53
01-bb	0.32	20.22	74.77	78.29	22.76	595.95	33713.13

matrix	M06-D7	M06-D8	M06-D9	M06-D10	M06-D11
size	294480x862290	862290x1395840	1274688x1395840	616320x1274688	122880x616320
rank	247704	614586	??????	493432	122879
Gauß	11878.27	89343.05	MT	47761.82	1463.77
Wiedemann	507067.62	?	?	?	123999.89

[Murri 2009]

Homology of the moduli space of smooth algebraic curves  $M_{g,n}$  using the Penner fatgraph complex. Since there is no canonical labeling of the graphs, the matrices are well-defined up to a permutation of rows and columns. The matrix labeled "M $g,n$ -D $k$ " relates graphs having  $k+1$  edges with graphs having  $k$  edges

# Gröbner Matrices

Matrix	Robot24_m5	Rkat7_m5	F855_m9	Cyclic8_m11
# rows	<b>404</b>	<b>694</b>	<b>2456</b>	<b>4562</b>
Non-zero	<b>15118</b>	<b>38114</b>	<b>171214</b>	<b>2462970</b>
Gauß	<b>0.52</b>	<b>1.84</b>	<b>10.54</b>	<b>671.33</b>
Wiedemann	<b>1.52</b>	<b>8.72</b>	<b>176.61</b>	<b>4138.77</b>

Sequential ranks modulo 65521 on a Sparc Ultra II 250 MHZ,  
arising during Gröbner bases computation [JC. Faugère]

Example:

8 variables system

5<sup>th</sup> matrix: 694×738

38114 non zero

Rank = 611

$$\begin{aligned}
 -x_1 + 2x_8^2 + 2x_7^2 + 2x_6^2 + 2x_5^2 + 2x_4^2 + 2x_3^2 + 2x_2^2 + x_1^2 &= 0 \\
 -x_2 + 2x_8x_7 + 2x_7x_6 + 2x_6x_5 + 2x_5x_4 + 2x_4x_3 + 2x_3x_2 + 2x_2x_1 &= 0 \\
 -x_3 + 2x_8x_6 + 2x_7x_5 + 2x_6x_4 + 2x_5x_3 + 2x_4x_2 + 2x_3x_1 + x_2^2 &= 0 \\
 -x_4 + 2x_8x_5 + 2x_7x_4 + 2x_6x_3 + 2x_5x_2 + 2x_4x_1 + 2x_3x_2 &= 0 \\
 -x_5 + 2x_8x_4 + 2x_7x_3 + 2x_6x_2 + 2x_5x_1 + 2x_4x_2 + x_3^2 &= 0 \\
 -x_6 + 2x_8x_3 + 2x_7x_2 + 2x_6x_1 + 2x_5x_2 + 2x_4x_3 &= 0 \\
 -x_7 + 2x_8x_2 + 2x_7x_1 + 2x_6x_2 + 2x_5x_3 + x_4^2 &= 0 \\
 -1 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1 &= 0
 \end{aligned}$$

# Combinatorics Matrices

Matrix	TF10	TF11	TF12	TF13	TF14	TF15	TF16	TF17	TF18	TF19
# rows	99	216	488	1121	2644	6334	15437	38132	95368	241029
Non-zero	622	1607	4231	11185	29862	80057	216173	586218	1597545	4370721
Gauß	0.01	0.02	0.22	4.37	61 .57	1002.14	18559.40	MT	MT	MT
Wiedemann	0.02	0.09	0.62	4.45	27.21	165.67	1248.46	7094.97	58893.6	359069

Sequential ranks modulo 65521 on a PIII 933 MHz, 1Gb

- Incidence matrices of unlabelled trees on n nodes versus unlabelled forests on n nodes with n-2 edges [N. Thiéry]
- Average number of non-zero elements per row growing from 6 to 18

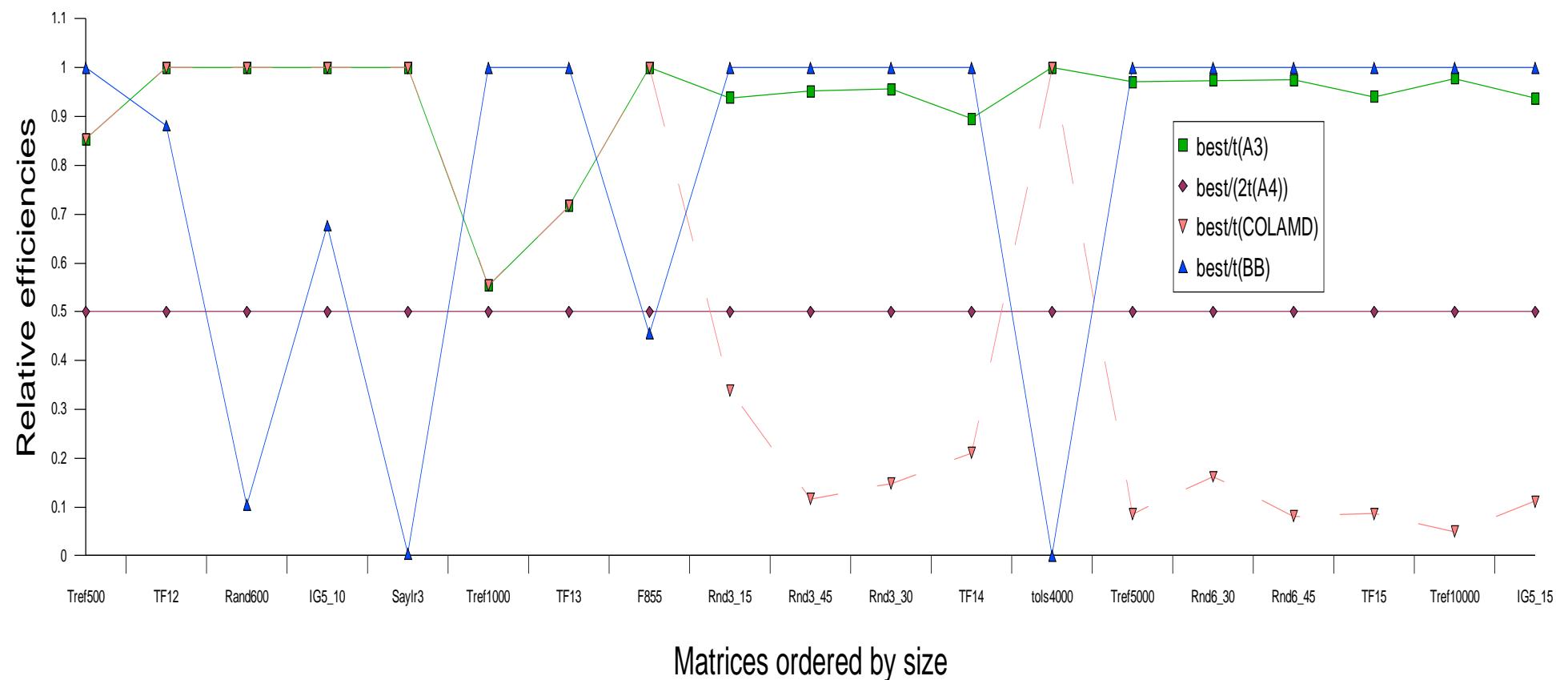
# **Wiedemann versus Gauß**

- Wiedemann is faster whenever  $6 < \omega < n/3$  ( $\approx 30\%$  sparse).
- Elimination with reordering is very efficient when matrix is:
  - Very small number of non-zero per row (no dense phase)
  - Quasi triangular (not much fill-in)
  - Very unbalanced (no dense phase)
- Iterative techniques are the only solution for the largest matrices (Memory Thrashing).  
⇒ Design of an adaptive heuristic

# Introspective rank algorithm

[Saunders-Wan 04]

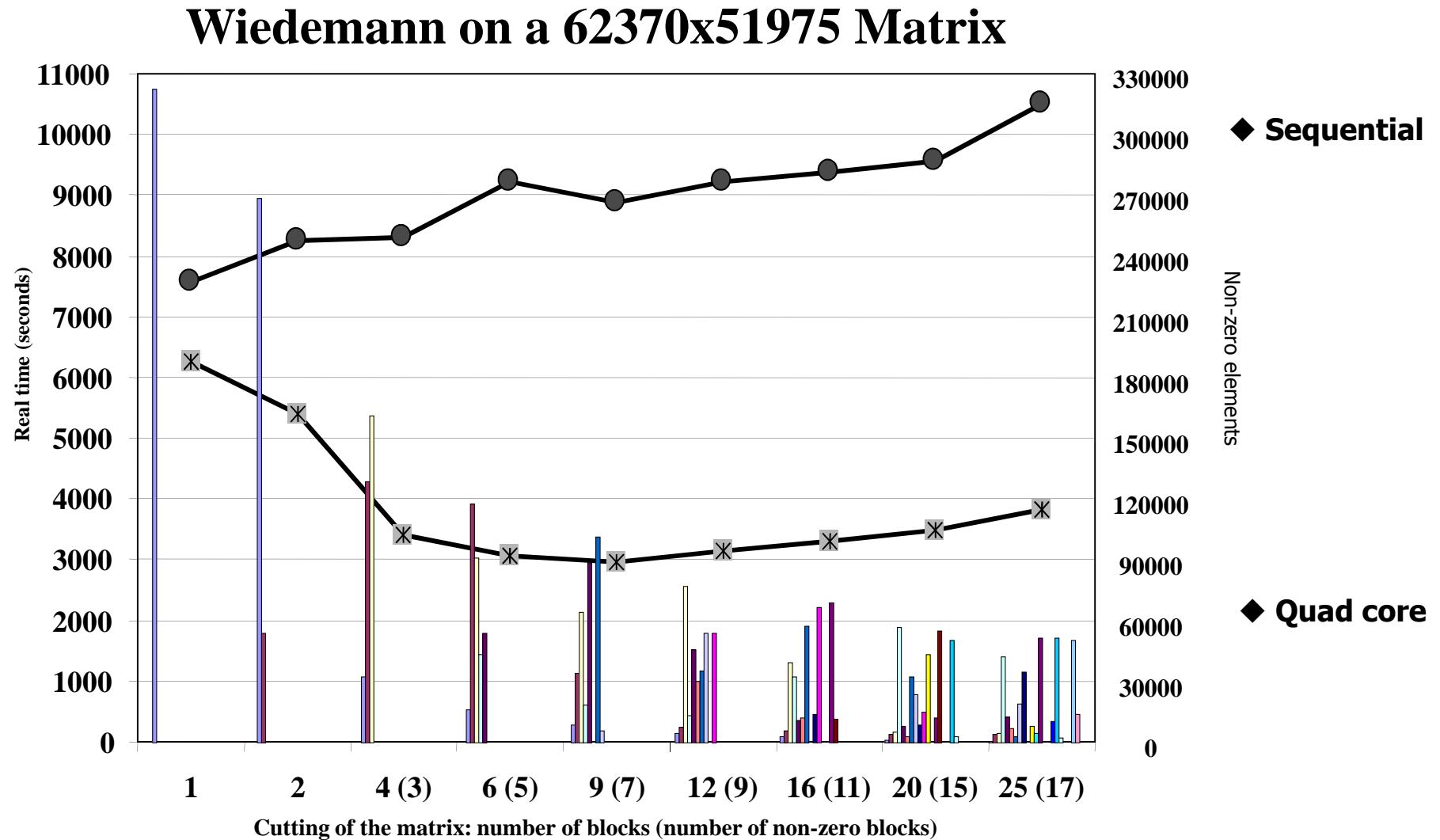
- Competition between elimination and black box algorithms
- Introspective eliminates until BB estimate is faster  
⇒ Always faster than the race



# **Contents**

1. Homology and linear algebra
2. Betti numbers
  - Rank via sparse elimination and reordering
  - Blackbox methods
  - Adaptive rank algorithm and domains of application
  - Parallel Wiedemann algorithm and scalability
3. Torsions
  - Sparse elimination with large precision integers and gcd's
  - Valence/Large invariant factor algorithm uses ranks
  - Adaptive Smith form

# Scalability: multicore algorithms



# Coppersmith's Parallel Block Wiedemann

- *[Coppersmith 94], [Kaltofen-Lobo 96], [Villard 97], ...*
- $p \times p$  matrix sequence  $U^T U, U^T A U, \dots, U^T A^k U$ 
  - Matrix-Vector products
  - FFLAS dot products to get the block sequence
  - Matrix generating polynomial of the block sequence
    - FFT-based  $\sigma$ -basis computation
  - Determinant of the obtained matrix-polynomial is the minimal polynomial
- Scalability *[D.-Elbaz-Giorgi-Urbanska 07]*
  - Sequential part, right now is  $\sigma$ -basis
  - E.g. for a  $1911130 \times 1955309$  matrix with 37M non-zero coefficients the different timings were:
    - 35 days on 50 processors for matrix-vector products
    - 1.56 CPU day for  $\sigma$ -basis 1 machine

# Contents

1. Homology and linear algebra
2. Betti numbers
  - Rank via sparse elimination and reordering
  - Blackbox methods
  - Adaptive rank algorithm and domains of application
  - Parallel Wiedemann algorithm and scalability
3. Torsions
  - Sparse elimination with large precision integers and gcd's
  - Valence/Large invariant factor algorithm uses ranks
  - Adaptive Smith form

# Naïve integer Smith form

$$\begin{pmatrix} 2 & 2 & 3 \\ 3 & 0 & -6 \end{pmatrix}$$

GCD

$$\begin{pmatrix} 2 & 2 & 1 \\ 3 & 0 & -6 \end{pmatrix}$$

Perm.

$$\begin{pmatrix} 1 & 2 & 2 \\ -6 & 3 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 2 \\ -6 & 3 & 0 \end{pmatrix}$$

Elim.  
column

$$\begin{pmatrix} 1 & 0 & 0 \\ -6 & 15 & 12 \end{pmatrix}$$

Elim.  
row

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 15 & 12 \end{pmatrix}$$

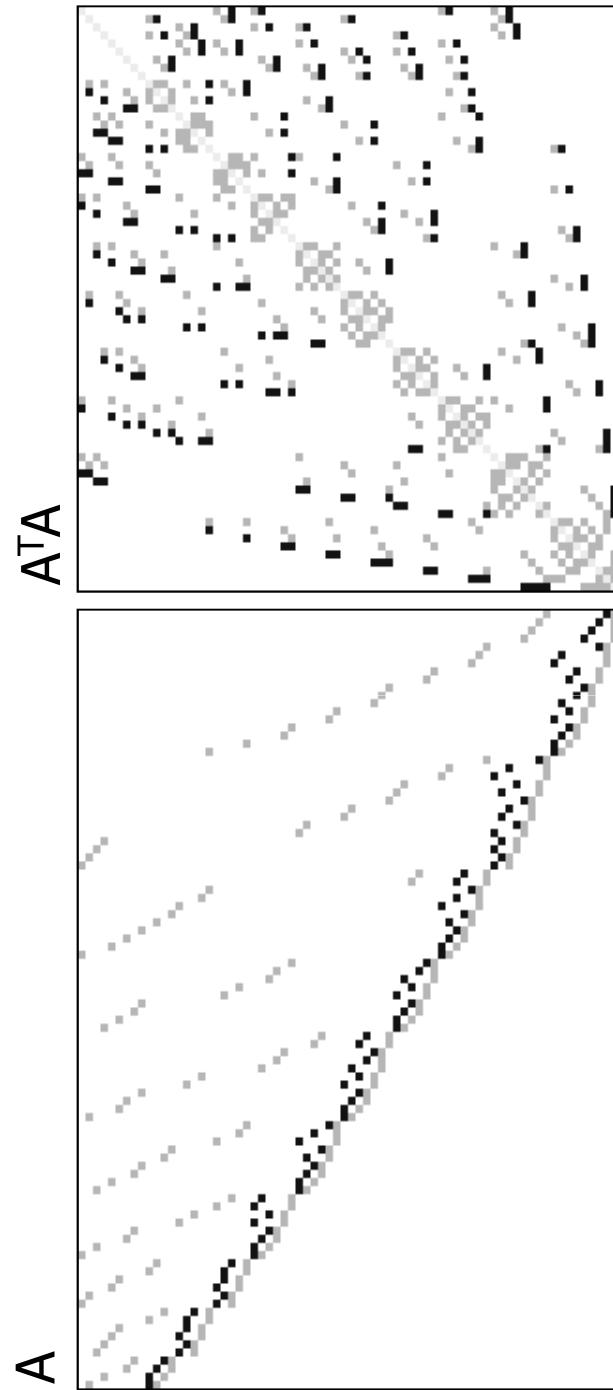
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 15 & 12 \end{pmatrix}$$

GCD

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 12 \end{pmatrix}$$

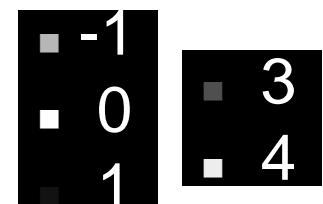
Elim.  
column

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix}$$



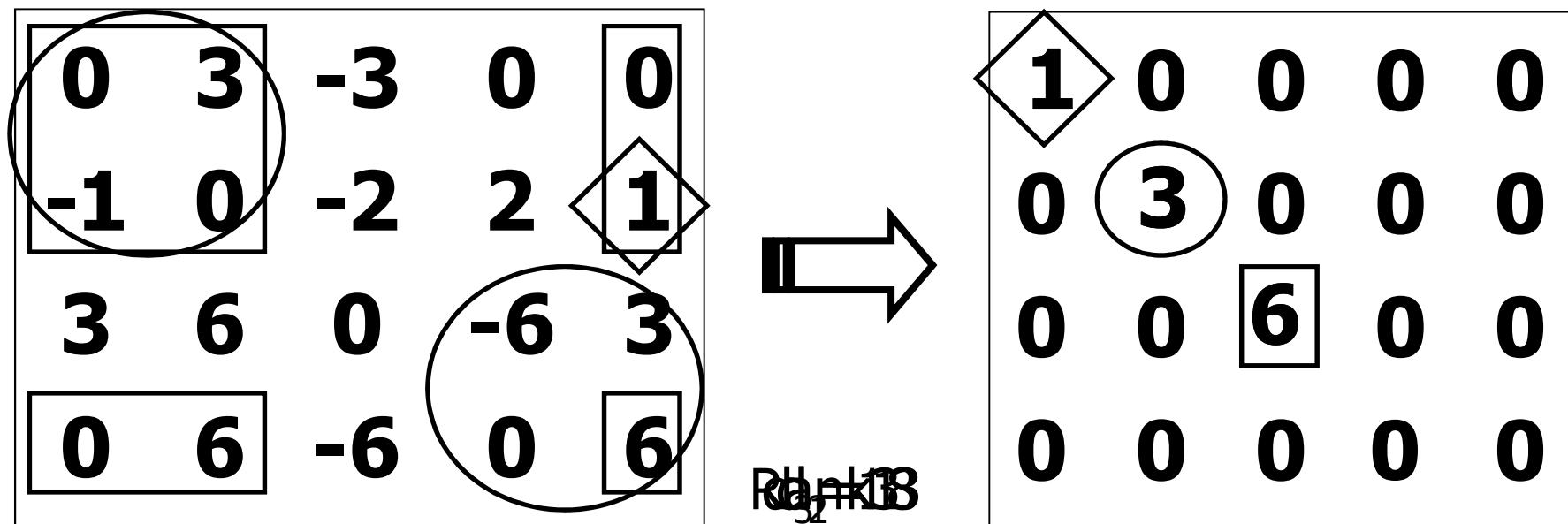
# Integer Smith form, ranks of Homology matrices

- Large dimensions
- Very sparse
- Small invariant factors
- Minimal polynomial of  $A^T A$  has very small degree



# Valence Integer Smith form

- $i^{\text{th}}$  Determinantal divisor:
  - $d_i = \text{GCD}(i \times i \text{ minors of } A)$
- $s_i = d_i/d_{i-1}$  is the  $i^{\text{th}}$  **Smith invariant** of  $A$



# Idea : work modulo powers of prime numbers

$\text{Rank}_q(A) = \text{largest } i \text{ such that } q \text{ does not divide } s_i$

When all the prime numbers appearing in the Smith form are known:  
compute the rank modulo powers of these primes

$$\begin{array}{ccccc} 0 & 3 & -3 & 0 & 0 \\ -1 & 0 & -2 & 2 & 1 \\ 3 & 6 & 0 & -6 & 3 \\ 0 & 6 & -6 & 0 & 6 \end{array} \sim \begin{array}{ccccc} -1 & 0 & -2 & 2 & 1 \\ 0 & 3 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

$\text{Rank}(A) \bmod 7 = 3 = \text{rank}(A) \text{ over } \mathbb{Z}$

$\text{Rank}(A) \bmod 2 = 2$

$\text{Rank}(A) \bmod 4 = 3$

$\text{Rank}(A) \bmod 3 = 1$

$\text{Rank}(A) \bmod 9 = 3$

$$\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

# **Valence Algorithm idea**

[D-Saunders-Villard 2000]

- 1 Either
  - Valence computation (valuation coefficient of the int. min. poly.)
  - Largest invariant factor computation
- 2 Set of prime numbers: L as an integer (only partial factorization)
- 3 Rank over  $\mathbb{Z}$ 
  - $q \notin L$ ,  $r = \text{rank}(A) \bmod q \Rightarrow$  there are r non zero invariants
- 4 Non-trivial invariant factors
  - $\forall p \in L$ , compute  $S_p$ , the local Smith form for powers of p.  
 $\Rightarrow$  Using ranks modulo  $p^k$
- 5  $S = \bigotimes S_p$  is the integer Smith form of A

# Ranks modulo non primes

- Smooth part: Powers of small prime powers  $p^k$ 
  - Sparse elimination:
    - use non-zero divisors as pivots
    - Then divide out by  $p$  and continue
  - Reeds-Sloane extends Berlekamp-Massey over finite rings
- Rough part: use dynamic evaluation
  - Use the field algorithm modulo a composite number
  - Failure means division by zero divisor
    - Composite number is factored
    - Split computation

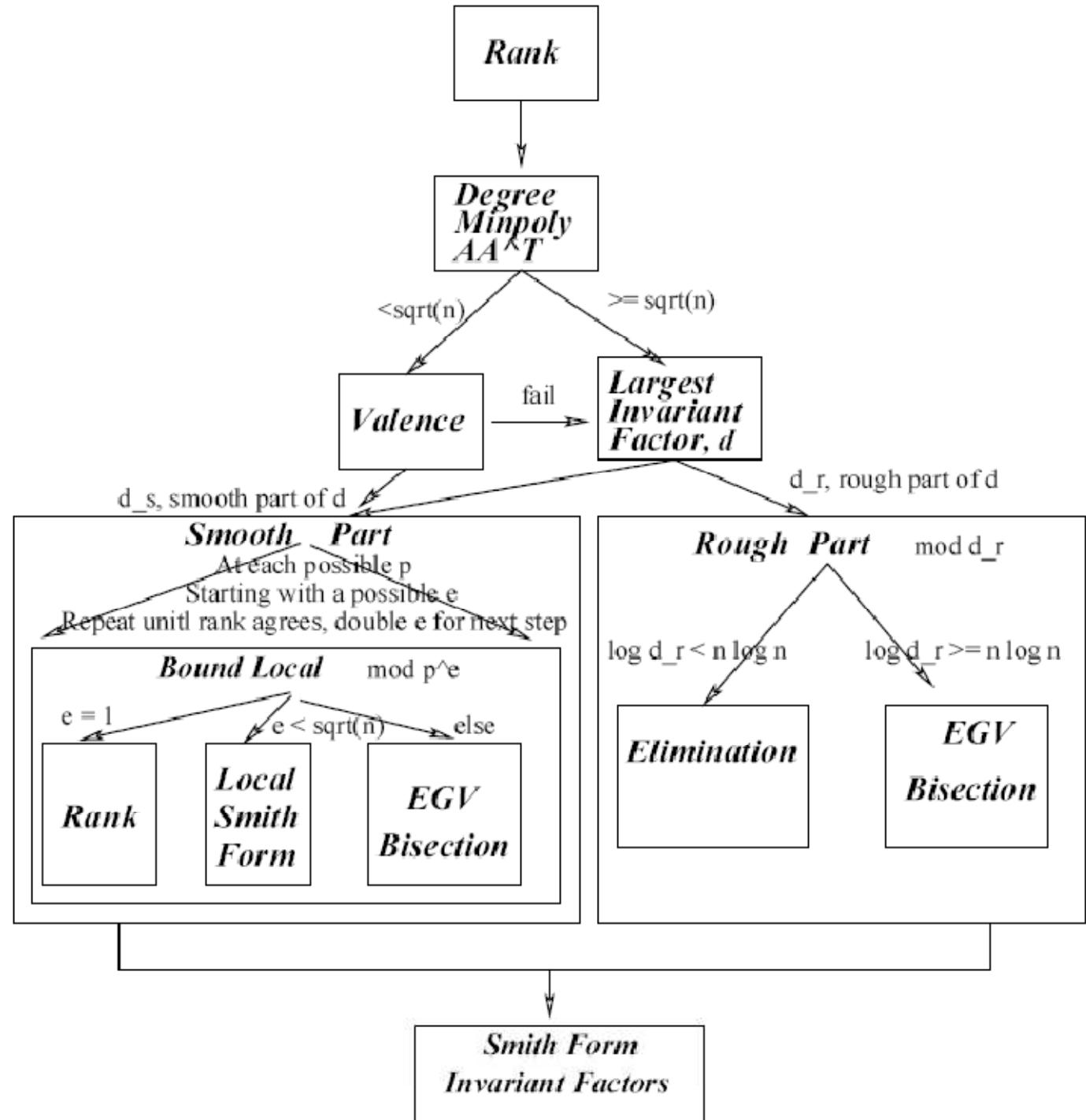
# Adaptive Smith form

[Saunders 2005]

1. Start with an introspective rank computation
2. Adaptively compute
  - a. the valence or
  - b. rough part of the largest invariant factor
    - ⇒ If the latter adaptively use a numeric-symbolic solver or tuned Dixon's lifting
    - ⇒ Bonus, also get rough part of several invariant factor
3. In most cases finish smoothly with local elimination and ranks for few tiny primes
4. In some cases also eliminate with a rough modulus
5. In rare cases of enormous rough part conduct Villard's binary search roughly

# Adaptive Smith form

[Saunders – Wan 2005]



# Some valence Smith form computations using 2 xeon 4x3.6 GHz, 6GB and 2 ia64 1.2GHz, 4GB

Matrix	Dim	Non-zero / row	Valence	Smith	Parallel time
mk13.b5	135135 x270270	4	$2^{10} \cdot 3^5 \cdot 5^2 \cdot 11 \cdot 13$	133991 ones 220 threes	14.75 hours on 6 processors
ch7-8.b5	141120 x141120	6	$2^{18} \cdot 3^9 \cdot 5^4 \cdot 7^3 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	92916 ones 35 threes 8 sixes	3.45 hours on 10 processors
ch8-8.b5	564480 x376320	6	$2^{21} \cdot 3^9 \cdot 5^4 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19$	276030 ones 1 three	95 hours on 9 processors

- ☺ Some Integer Smith normal forms were only partial *[D, Saunders, Villard, 2000]*
  - ☺ We might have had larger powers of 2 or 3
- ☺ With more memory (a machine with 128GB), sparse elimination mod  $p^k$  got the local Smith forms at 2 and 3 for all these matrices:
  - ☺ Result for ch7-8.b5, mk13.b5 and ch8-8.b5 are now proven

# Challenges

- Scalability
  - Take advantage of multi-core/GPU architectures
  - Block theory is there
  - Design efficient and reliable block algorithms for these architectures
- Adaptivity
  - Efficient Ranks
    - Adaptive sparse/dense elimination
    - Parallel/block/out-of-core sparse elimination
    - Parallel/block matrix-vector products for faster iterative methods (better suited to e.g. GPU's ?)
  - Efficient Smith form
    - Invariant factors
    - Full BB local Smith form

# **IMPLEMENTATIONS**

**SimpHom package for GAP**

**C++ LinBox-1.1.6**

**Within Sage**

Bellamy Botting Boyer Caviness Chen Devore Dumas Duran Eberly Fendt  
Gautier Giesbrecht Giorgi Gold Hovinen Kaltofen Karel Lee Lobo May  
Morozov Pernet Pritchard Reichard Roch Roche Saunders Schrag Seagraves  
Steijn Storjohann Turner Urbanska Villard Wan Xiang Youse Yuhasz